SigIMS Software Users Guide

- Kamailio
- rtpproxy
- Libnetfilter
- Encryption engine

Signalogic, Inc.
9617 Wendell
Dallas, TX
75243

©Signalogic, Inc
Mar 2011, Rev F3

Table of Contents

1 SYSTEM OVERVIEW 4

1.1 Hardware System Architecture 4
1.2 Software System Architecture 7
1.3 Call Flow/Message Flow Diagrams 8

2 INSTALLATION 10

2.1 Installing Kamailio 10
2.1.1 Installing Kamailio 1.5x 10
2.1.2 Installing Kamailio v3.0.x 12
2.2 Installing rtpproxy v1.2.1 15
2.3 Installing Libnetfilter 16
2.4 Installing the Encryption Engine 18
2.5 Adding Signalogic Software Modifications to rtpproxy and Kamailio 19

3 SIGNALOGIC SOFTWARE 22

3.1 Compiling and Running the Encryption Engine 22

4 TEST CALL SETUP 24

4.1 Scenario 1 (X-lite to X-lite) 24
4.2 Scenario 2 (X-lite to Mobile Dialer) 26

APPENDIX A, Kamailio Configuration File

APPENDIX B, Enforcing Independent control over encryption of sip, media

TABLE OF FIGURES

Figure 1 : System Interconnect Architecture 4
Figure 2 : System Software Architecture 5
Figure 3 : RTP Packet Flow 6
Figure 4 : Software System Call Flow 7
Figure 5 : Register Event 8
Figure 6 : Call Flow Events 9
Figure 7 : nfqnl_test output screenshot 17
Figure 8 : X-lite softphone settings 24
Figure 9 : X-lite softphone settings 25
Figure 10 : X-lite softphone settings 26
Figure 11 : Mobile dialer account configuration screens 27
Figure 12 : Asterisk sip.conf screenshot 28

1 System Overview

This chapter describes the overall system architecture. It explains overall system interconnect architecture between Kamailio, Asterisk and rtpproxy. This chapter also explains various data packet flow and message exchange diagrams between two endpoints through Asterisk, Kamailio and rtpproxy.

1.1 Hardware System Architecture

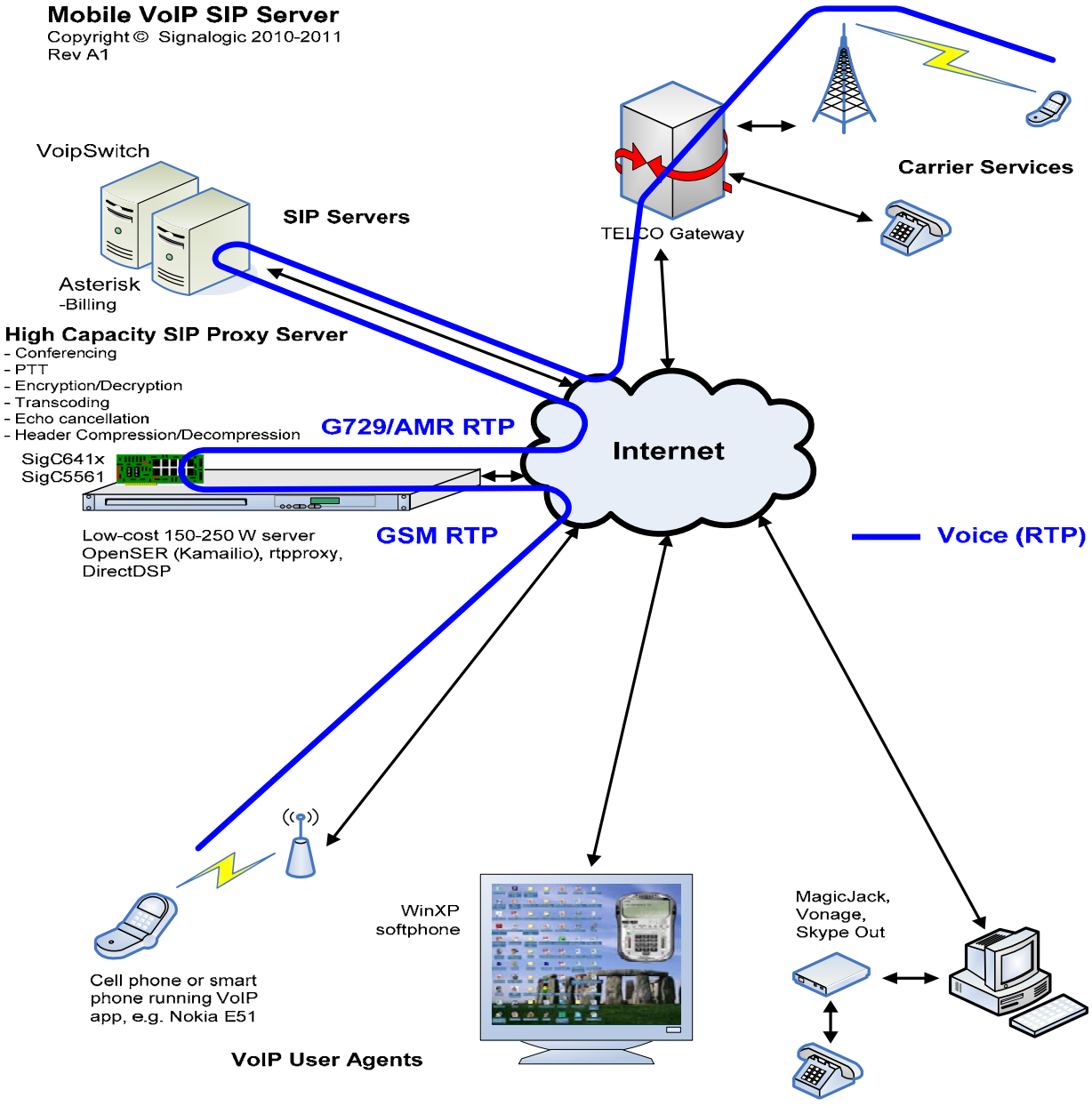

Figure 1: System Interconnect Architecture

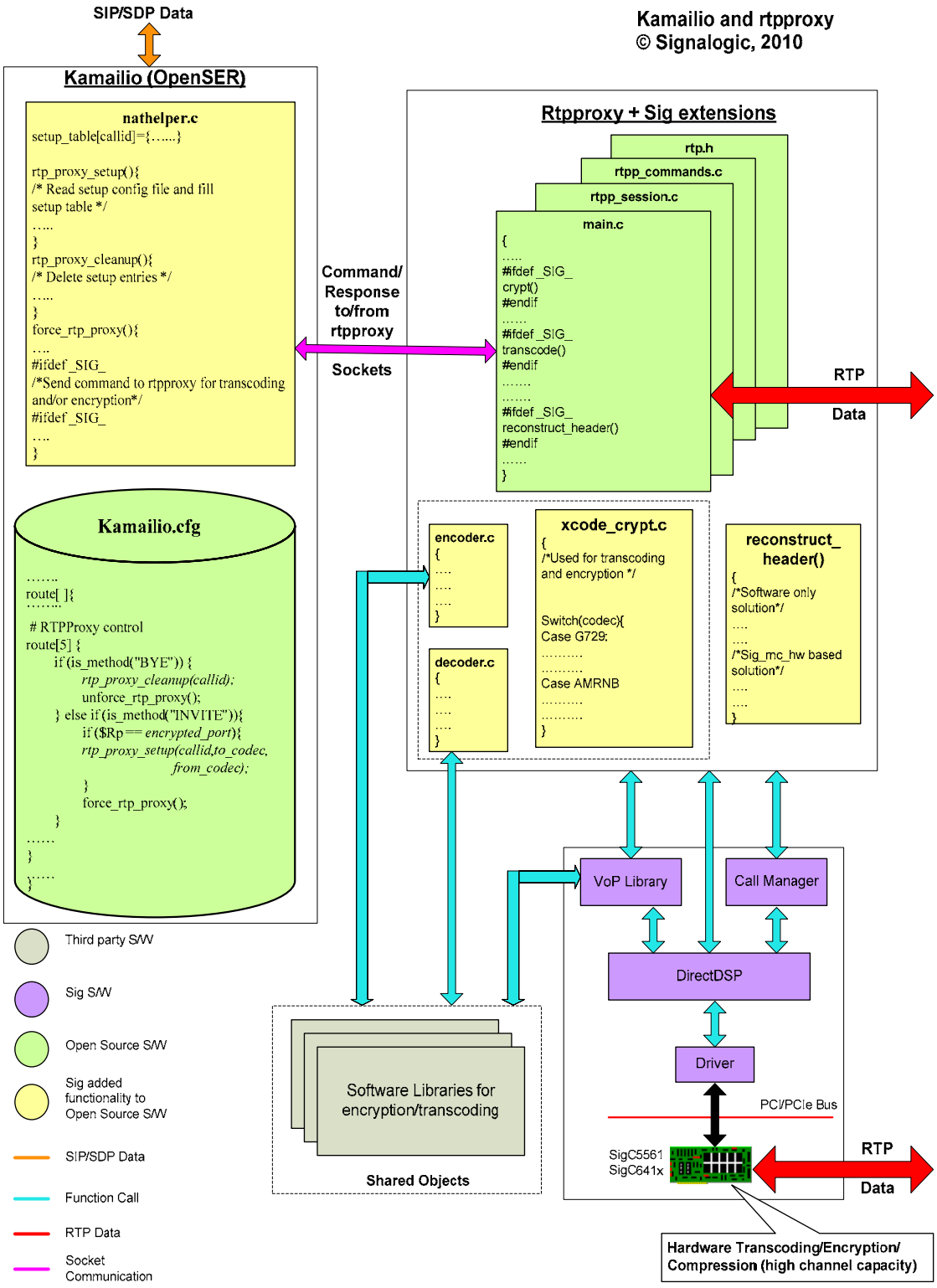

Figure 2: System Software Architecture

Figure 1 illustrates the overall system interconnect architecture. As seen in the Figure, rtpproxy and Kamailio reside on the server containing voice and/or video cards, while a separate server is dedicated to Asterisk. All of them are connected them to the internet. The blue line represents the path taken by voice data between two endpoints. Kamailio acts as a

SIP proxy, while Asterisk acts as SIP registrar and is responsible for various voice services like voicemail, billing, etc. rtpproxy, as the name suggests, acts at RTP proxy server.

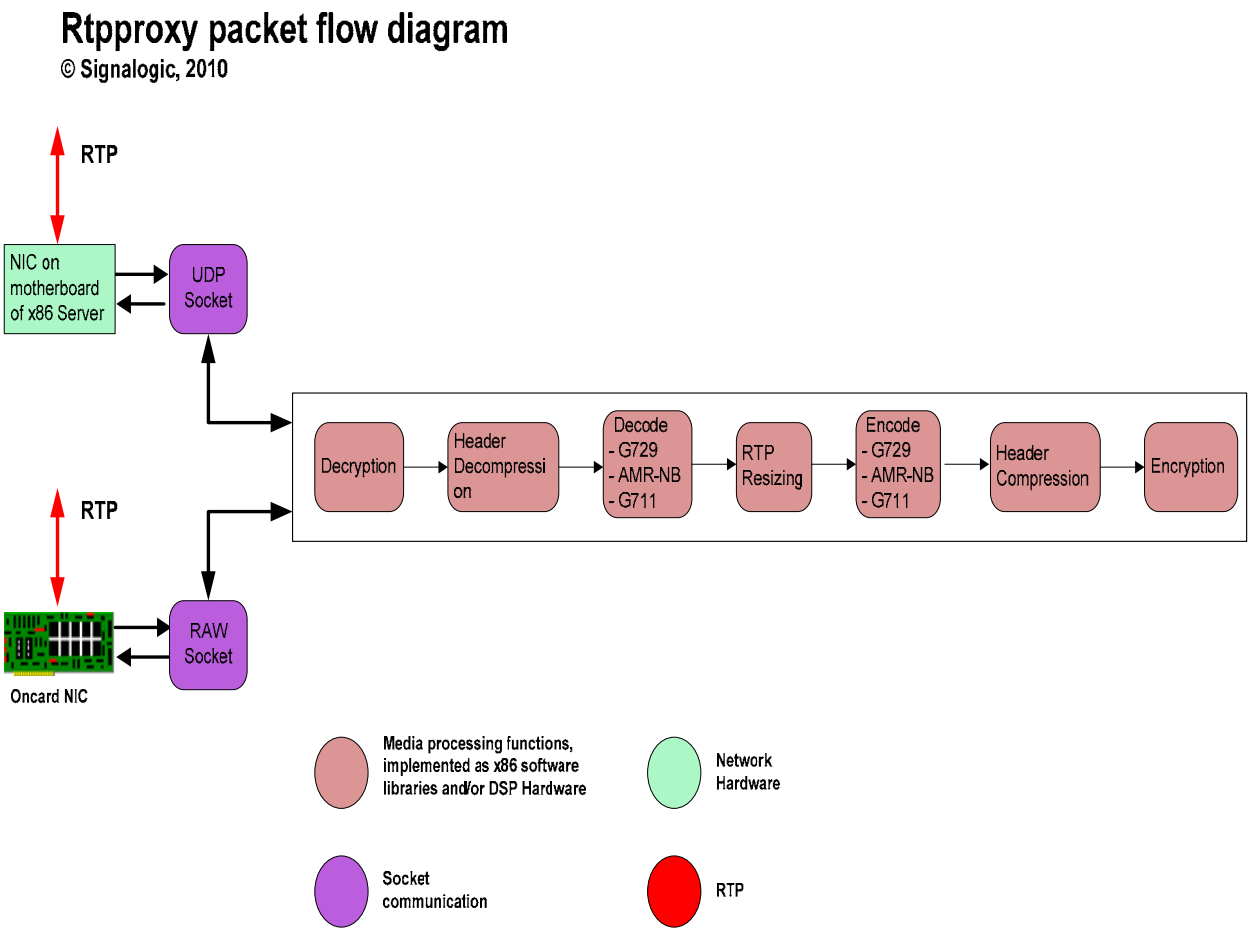

Figure 3: RTP Packet Flow

Figure 3 represents the RTP packet flow inside rtpproxy.

1.2 Software System Architecture

Figure 2 represents software interconnect architecture involving Kamailio, rtpproxy and DirectDSP. It shows the logical data/message connections between them.

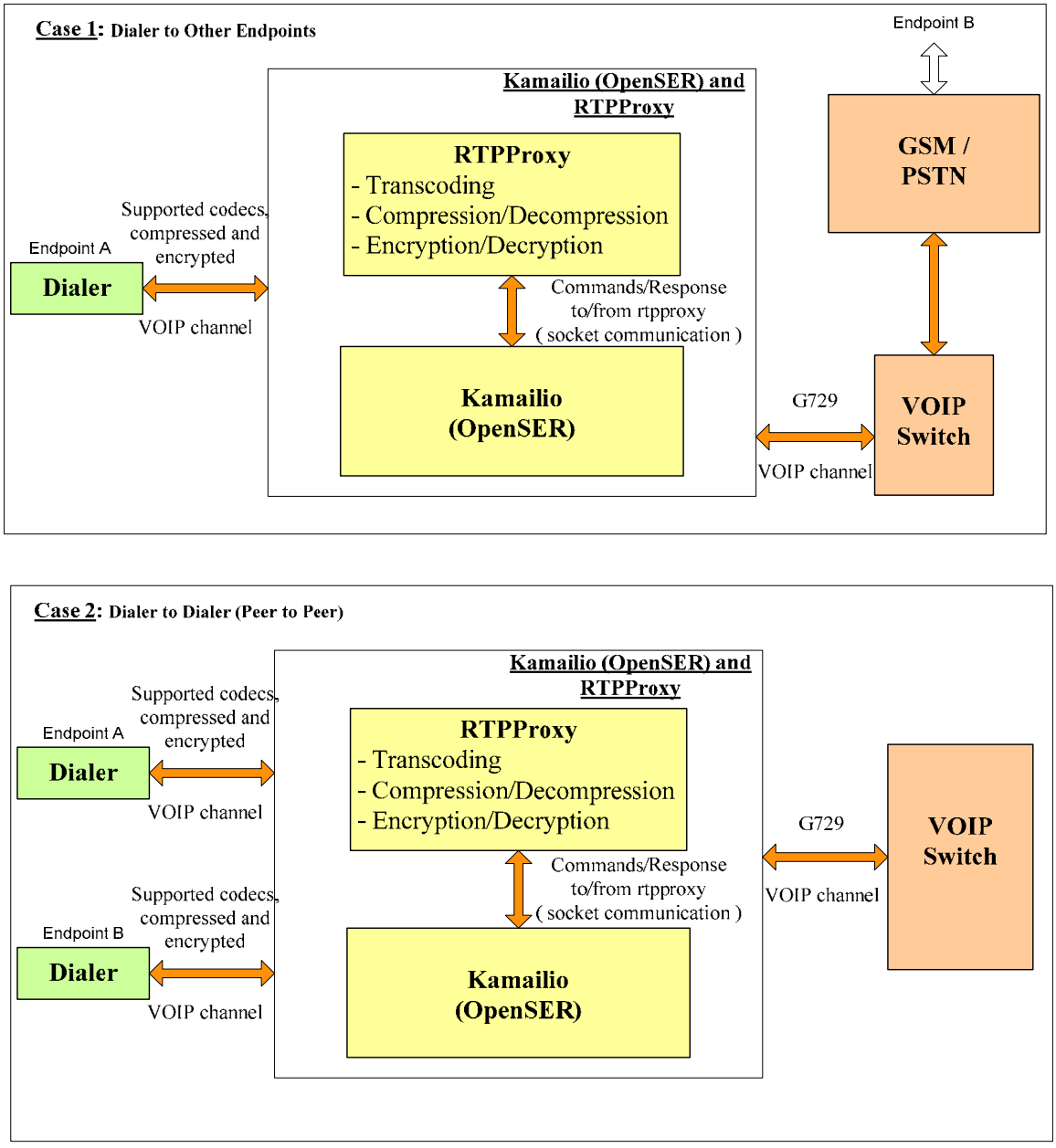

Figure 4: Software System Call Flow

Figure 4 represents the systemic software call flow. It demonstrates how various softwares are integrated together to demonstrate the software call flow. Here two possible cases are illustrated. In the first case, the VOIP dialer application is used to establish a session (call) with either cell phone or landline. In the second case, both the endpoints support the VOIP dialer application.

1.3 Call Flow/Message Flow Diagrams

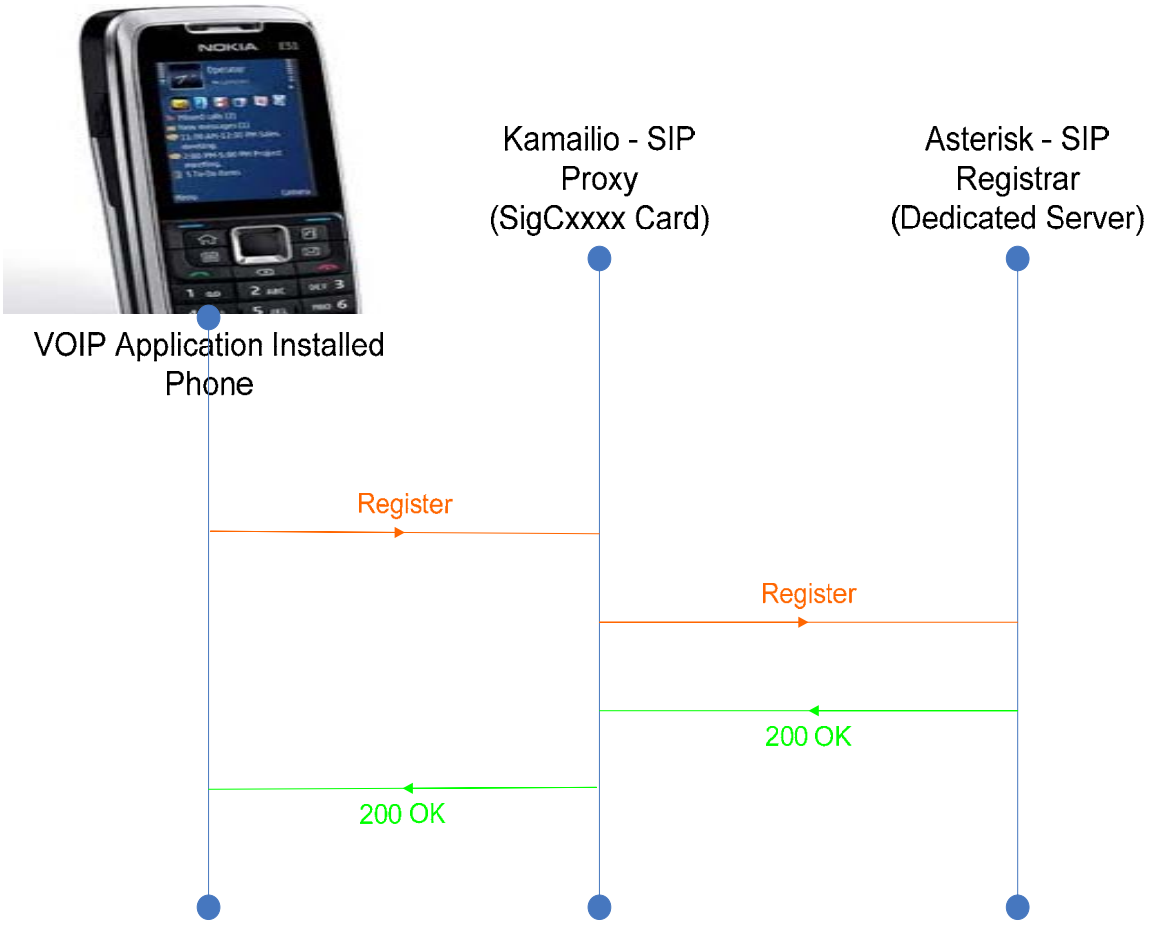

Figure 5: Register Event

The diagram shown in Figure 5 represents the register event. This event takes place when the VOIP dialer application registers itself to the SIP registrar (i.e. Asterisk). Each SIP message has to pass through a SIP Proxy in this case. So, a register message method is issued by the VOIP dialer to the SIP Proxy and the SIP Proxy forwards it to the SIP Registrar.
As soon as, the register event is verified by the SIP Registrar, it sends back a '200 OK' to the Proxy, this in turn is returned back to the dialer. The dialer application now binds a port on the Registrar with a particular username and phone number.

Figure 6 illustrates various events/messages generated when a session is initiated, established from the one end point (dialer application) to other endpoint (cell phone). An invite event (SIP event) along with the session data (in SDP) is generated to invite an endpoint to participate in

a session. The invite passes through the Proxy to the Asterisk, which passes it to the telecommunications gateway and onto the cell phone.

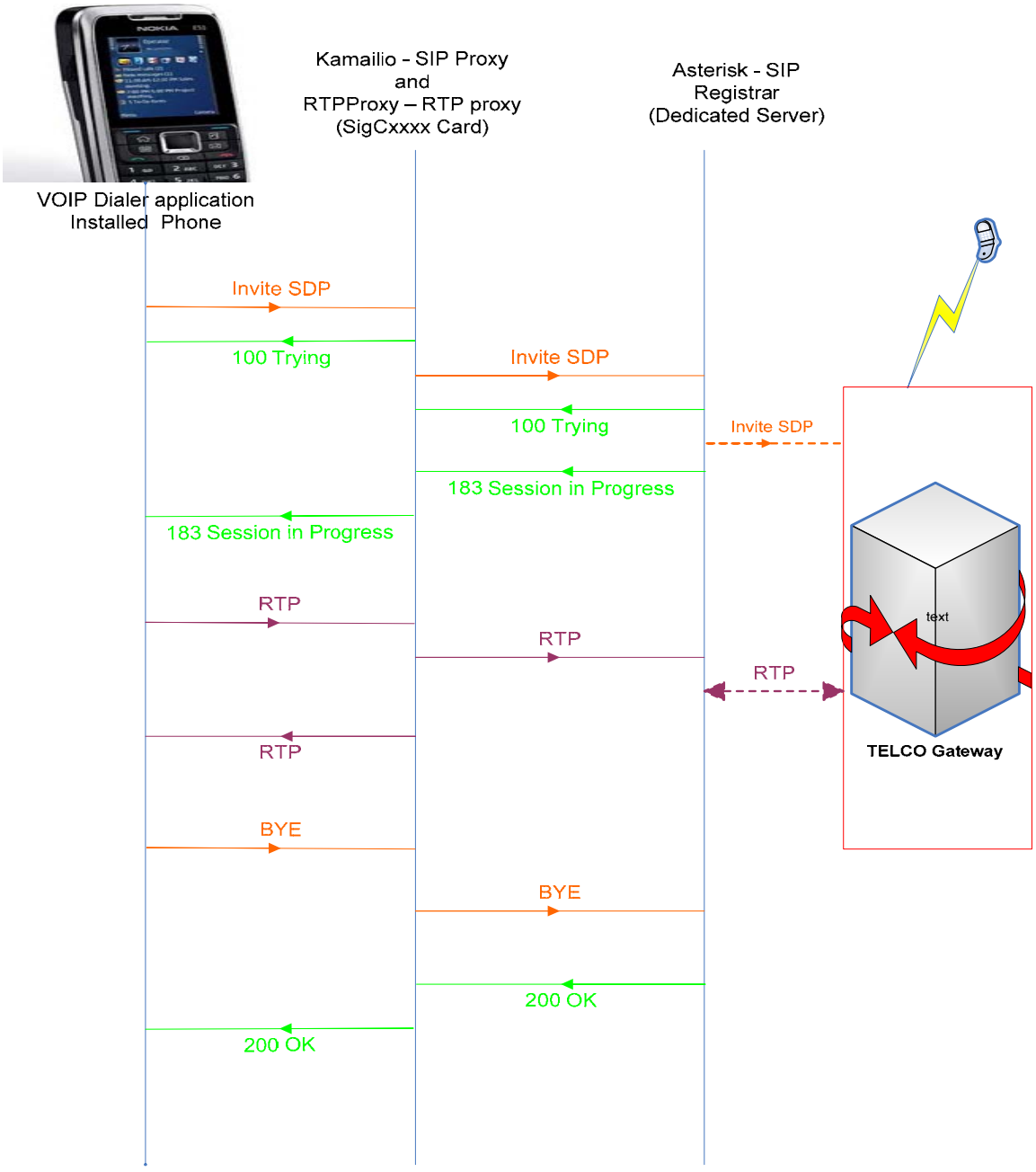

Figure 6: Call Flow Events

As soon, the phone is answered a '183 Session in progress' message is send back to the dialer. An RTP session is begun. Since, RTP proxy is installed on the same SigCxxxx card. The RTP traffic then passes through to the SigCxxxx card - Asterisk and on to either endpoint.

To end the session, either endpoint can send a 'BYE' event, if the other endpoint does not have any more data to transmit it replies with '200 OK' and the current session is ended.

2 Installation

This chapter describes the various softwares, prerequisites and their installation procedures, including Kamailio, rtpproxy, Libnetfilter and Encryption engine.

2.1 Installing Kamailio

2.1.1 Installing Kamailio 1.5x

Prerequisites

To be able to follow the guidelines from this document you need **root** access.
The following packages are required before proceeding to the next steps.
- *svn* client: yum install subversion
- *gcc* compiler: yum install gcc
- *flex* – yum install flex
- *bison* – yum install bison
- *libmysqlclient15-dev* - yum install libmysqlclient15-dev
- *make* – yum install make

Getting sources from SVN

First of all, you have to create a directory on the file system where the sources will be stored.
mkdir -p /usr/local/src/kamailio-1.5.0
cd /usr/local/src/kamailio-1.5.0

Download the sources from svn using the following command.
svn co http://openser.svn.sourceforge.net/svnroot/openser/branches/1.5 sip-server

Tuning Makefiles

Next step is to edit Makefile.var files to include the MySQL module.
cd sip-server
nano Makefile.vars

Uncomment the next line in the file:
MODS_MYSQL=on
Save the Makefile.vars and exit.

Note: if you need PCRE-dependent modules (e.g., lcr, dialplan, regexp), install the PCRE devel library and uncomment
MODS_PCRE=on

Compiling Kamailio

Once the mysql module was removed from excluded modules list, you can compile Kamailio.

make all

You can get a nicer output of the compilation process using:
NICER=1 make all

Install Kamailio

When the compilation is ready, install Kamailio with the following command:
make install

Edit configuration file

To fit your requirements for the VoIP platform, you have to edit the configuration file.
/usr/local/etc/kamailio/kamailio.cfg

Note:

(a) Kamailio needs to be configured to listen on the port that carries encrypted SIP, i.e. the following line must be included in addition to other '*listen=*' lines
listen=udp:server_ip_address:encrypted_port_num

(b) Kamailio can be started up using the init.d script /etc/init.d/kamailio start

(c) Uncomment relevant sections of Kamailio.cfg to enable Registrar/Proxy functionality. Example configuration file available in VSS.

(d) rtpproxy must be started before starting Kamailio.

(e) To Stop kamailio type
ps –ef | grep kamailio → This will display all the processes created by kamailio, to stop kamailio, there very first kamailio process (root) has to be killed

To kill kamailio use ' kill *pid* ', pid is the number of the first process.

(f) To start and register logging process for Kamailio, follow the following steps
- Before starting Kamailio, create a file *kamailio_log.log* in */var/log/*
- Add the following lines to */etc/syslog.conf*
  Local0. /var/log/kamailio_log.log*
- Restart the logging service by typing
  $ /etc/init.d/syslog restart

2.1.2 Installing Kamailio v3.0.x

Prerequisites

To be able to follow the guidelines from this document you need **root** access.
The following packages are required before proceeding to the next steps.
- *git* client: apt-get install git-core - it is recommended to have the latest version, which might not be part of the distribution yet, but you can get it from: http://git-scm.com/
- *gcc* compiler: apt-get install gcc
- *flex* - apt-get install flex
- *bison* - apt-get install bison
- *libmysqlclient15-dev* - apt-get install libmysqlclient15-dev
- *make* - apt-get install make
- *libcurl* - apt-get install libcurl4-openssl-dev
- *libxml2* - apt-get install libxml2-dev
- *libpcre3* - apt-get install libpcre3-dev

Getting sources from GIT

First of all, you have to create a directory on the file system where the sources will be stored.
mkdir -p /usr/local/src/kamailio-3.0.0
cd /usr/local/src/kamailio-3.0.0

Download the sources from GIT using the following commands.
git clone --depth 1 git://git.sip-router.org/sip-router kamailio
cd kamailio
git checkout -b kamailio_3.0 origin/kamailio_3.0

Compile Kamailio (Openser)

Go to source directory where Kamailio is downloaded i.e. */usr/local/src/kamailio-3.0.x*
To install all the modules
make all

You can get a nicer output of the compilation process using:
make Q=1 all

Installing Kamailio (OpenSER)

When the compilation is ready, install Kamailio (OpenSER) with the following command:
make install

What and where Kamailio was installed

The binaries and executable scripts were installed in:
/usr/local/sbin

These are:
kamailio - Kamailio (OpenSER) server
kamdbctl - script to create and manage the Databases
kamctl - script to manage and control Kamailio (OpenSER) server
sercmd - CLI - command line tool to interface with Kamailio (OpenSER) server

To be able to use the binaries from command line, make sure that '*/usr/local/sbin*' is set in *PATH* environment variable. You can check that with '*echo $PATH*'. If not and you are using '*bash*', open '*/root/.bash_profile*' and at the end add:
PATH=$PATH:/usr/local/sbin
export PATH

Kamailio (OpenSER) modules are installed in:
/usr/local/lib/kamailio/modules/
/usr/local/lib/kamailio/modules_k/

The documentation and readme files are installed in:
/usr/local/share/doc/kamailio/

The man pages are installed in:
/usr/local/share/man/man5/
/usr/local/share/man/man8/

The configuration file was installed in:
/usr/local/etc/kamailio/kamailio.cfg

Edit configuration file

To fit your requirements for the VoIP platform, you have to edit the configuration file.
/usr/local/etc/kamailio/kamailio.cfg

Maintenance

The maintenance process is very simple right now. You have to be user 'root' and execute following commands:
cd /usr/local/src/kamailio-3.0.0/kamailio
git pull origin
make all
make install
/etc/init.d/kamailio restart
Now you have the latest Kamailio (OpenSER) v3.0.0 running on your system.

Time for maintenance service

Notification about GIT commits are sent to the mailing list: **sr-dev@lists.sip-router.org**. Each commit notification contains the reference to the branch where the commit has been done. If the commit message contains the lines:
Module: sip-router
Branch: kamailio_3.0
then an update has been made to Kamailio (OpenSER) version 3.0.x and it will be available to the public GIT in no time.

Reference Webpage: http://www.kamailio.org/dokuwiki/doku.php/install:kamailio-3.0.x-from-git

Notes

(a) Kamailio needs to be configured to listen on the port that carries encrypted SIP, i.e. the following line must be included in addition to other '*listen=*' lines
listen=udp:server_ip_address:encrypted_port_num

(b) Kamailio can be started by command '*kamailio*', only once the PATH variable is updated

(c) rtpproxy must be started before starting Kamailio.

(d) To stop Kamailio type
ps –ef | grep kamailio → This will display all the processes created by Kamailio, to stop Kamailio, there very first Kamailio process (root) has to be killed

To kill kamailio use '*kill pid* ', *pid* is the number of the first process.

(e) To start and register logging process for Kamailio, follow the following steps
- Before starting Kamailio, create a file *kamailio_log.log* in */var/log/*
- Add the following lines to /etc/syslog.conf
  Local0. */var/log/kamailio_log.log*
- Restart the logging service by typing
  $ /etc/init.d/syslog restart

2.2 Installing rtpproxy v1.2.1

First download rtpproxy stable version 1.2.1 from http://www.rtpproxy.org/

Navigate to the directory containing the downloaded tarball and follow the instruction below

$ tar -xvf rtpproxy-1.2.1.tar.gz
$ cd rtpproxy-1.2.1
$ ./configure
$ make
$ make install

Once rtpproxy has been installed it can be stared using the following command
$ rtpproxy –l server_ip_address –s udp:127.0.0.1:port_number –F –d DBUG:LOG_LOCAL5

Notes

(a) *server_ip_address* is the ip address of the proxy server

(b) *port_number* is the port number that is defined in the line
modparam("nathelper","rtpproxy_sock","udp:127.0.01:*port_number*")
in the configuration file for kamailio.

(c) To stop rtpproxy type
ps –ef | grep rtpproxy → This will display the process created by rtpproxy, to stop rtpproxy, the rtpproxy process (root) has to be killed

To kill rtpproxy use '*kill pid* ', *pid* is the number of the first process.

To start and register logging process for rtpproxy, follow the following steps
- Before starting rtpproxy, create a file *rtpplog.log* in */var/log/*
- Add the following lines to */etc/syslog.conf*
  local5. */var/log/rtpplog.log*
- Restart the logging service by typing
  $ /etc/init.d/syslog restart

2.3 Installing Libnetfilter

Prerequisites

Linux kernel that includes nfnetlink_queue subsystem (2.6.14 or greater)

Installing nfnetlink and netfilter_queue

Download libnfnetlink v1.0.0 from *http://ftp.netfilter.org/pub/libnfnetlink/*
- Navigate to the directory that contains the tarball that has been downloaded and follow instructions below

$ tar xvfj libnfnetlink-1.0.0.tar.bz2
$ cd libnfnetlink-1.0.0/
$ ./configure --prefix=/usr
$ make
$ make install

Download libnetfilter_queue v0.0.17 *http://ftp.netfilter.org/pub/libnetfilter_queue/*
- Navigate to the directory that contains the tarball that has been downloaded and follow instructions below

$ tar xvfj libnetfilter_queue-0.0.17.tar.bz2
$ cd libnetfilter_queue-0.0.17
$ ./configure --prefix=/usr
$ make
$ make install

Verifying install

Start iptables service by using the command
$ service iptables start

Add an iptables rule to test libnetfilter_queue :
$ iptables –t mangle –A INPUT –j NFQUEUE

Navigate to the directory where libnetfilter_queue has been installed
$ cd ./libnetfilter_queue-0.0.17/utils
Compile nfqnl_test.c :
$ gcc nfqnl_test.c –lnfnetlink –lnetfilter_queue

Run a.out : *./a.out*

Below is a screenshot of the expected output (Note: If there is nothing displayed, try opening a web browser session).

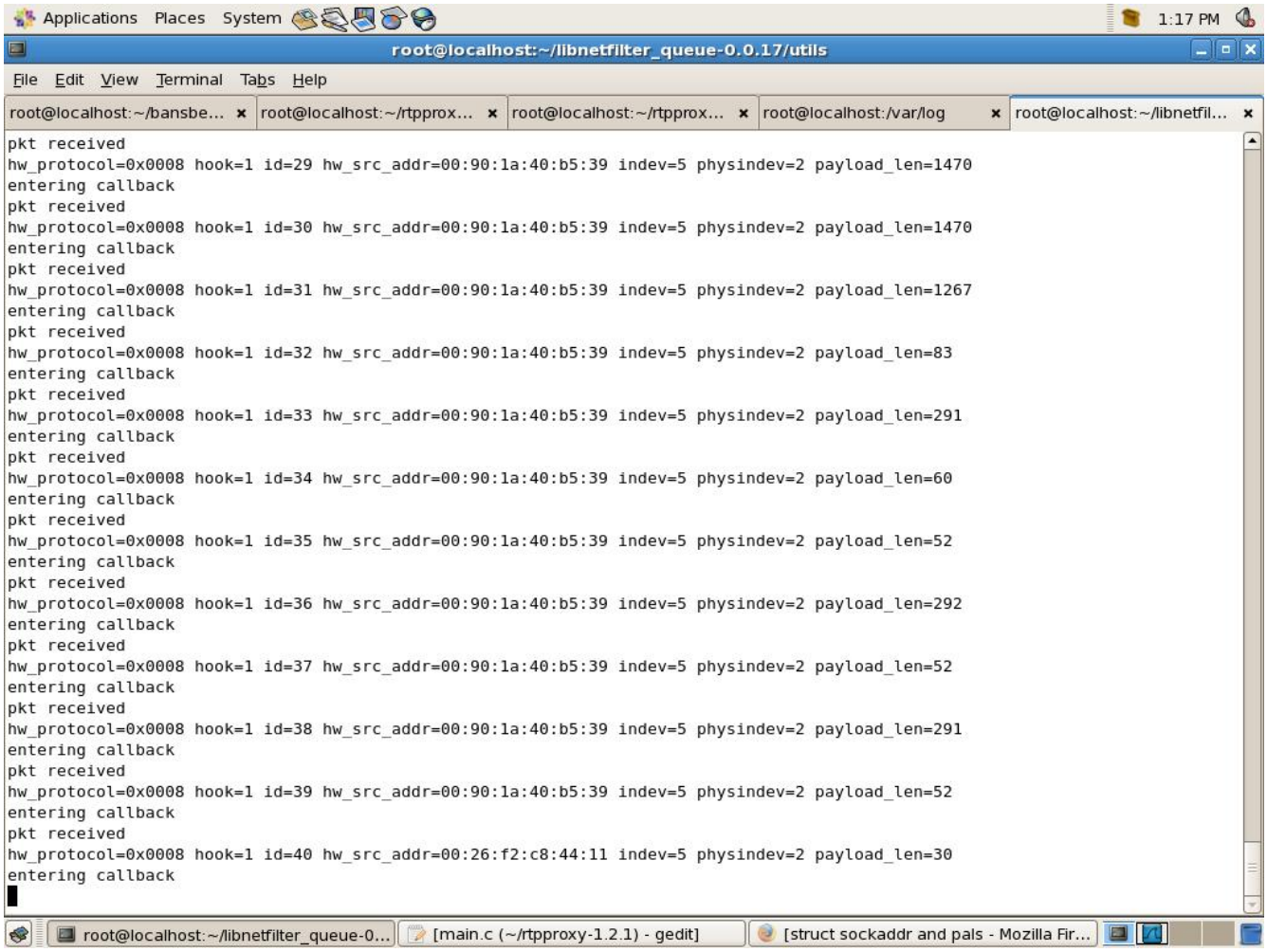

Figure 7 : Screenshot showing the output of nfqnl_test

2.4 Installing the Encryption Engine

- Start iptables service by using the command
  service iptables start

- Add the following iptables rules
  iptables –t mangle –A PREROUTING –p udp --dport port_num –j NFQUEUE
  iptables –t mangle –A OUTPUT –p udp --sport port_num –j NFQUEUE
  iptables –t mangle –A PREROUTING –p tcp --dport port_num –j NFQUEUE
  iptables –t mangle –A OUTPUT –p tcp --sport port_num –j NFQUEUE

- Navigate to the directory where encryption_engine.c has been copied

- Compile encryption_engine.c :
  gcc encryption_engine.c –o crypt_eng.out –lnfnetlink –lnetfilter_queue

Run crypt_eng.out :

./crypt_eng.out encryption_key sip_server_ip1 sip_server_ip2..

Notes

(a) encryption_key is the plain text key that is used during encryption

(b) sip_server_ipx is the dotted decimal notation IP address of all SIP servers in use

(c) port_num = encryption port number (i.e. 9090)

(d) To add additional ports for encryption, additional rules should be added to iptables by following the above model.

2.5 Adding Signalogic Software Modifications to rtpproxy and Kamailio

Modifying rtpproxy and rebuild

- Replace main.c, rtpp_session.h, rtpp_command.c with the files provided in Signalogic software package.

- Add reconstruct_header.h, reconstruct_header.c, rtplib_xced.c, rtplib_xced.h, rtplib_encoder.c, rtplib_decoder.c, rtplib.c and Makefile.in form Signalogic software package into rtpproxy root folder.

- To enable header compression add
  #define _SIG_HEADER_COMPRESSION in *main.c*

- To rebuild rtpproxy, cd into the rtpproxy root folder and follow the steps below
  $ ./configure
  $ make all
  $ make install

- To start rtpproxy in background use with Debug mode
  rtpproxy –l proxy_ip –s udp:127.0.0.1:7722 –F -_1 –d DBUG:LOG_LOCAL5

- To start rtpproxy in forground use with Debug
  rtpproxy –l proxy_ip –s udp:127.0.0.1:7722 –F -_1 –f –d DBUG

Modifying Kamailio and rebuild

- Replace nathelper.c under kamailio_install_folder/modules/nathelper, and rebuild Kamailio as per the instructions below
  $ make all
  $ make install
- Replace the original *kamailio.cfg (/root/usr/local/kamailio/kamailio.cfg)* the *kamailio.cfg* provided with the software
- Modifying Kamailio configuration file.

Features	Kamailio configuration file modification	Comments
Encryption	- On *route[5]* - In *is_method("INVITE")* function - On port *($Rp == 9090)* - Add *rtp_proxy_setup("")*above *'force_rtp_proxy()'* as shown in example cfg file in Appendix A - Under *onreply_route[1]'* - Add *rtp_proxy_setup("")*above *'force_rtp_proxy()'* as shown in example cfg file in Appendix A	- "encrytpionkey" is replaced by the actual encryption key used in the algorithm. - Use port 9090 to make encrypted calls - Comment out any other *rtp_proxy_setup()* function calls.
Transcoding	- On *route[5]* - In *is_method("INVITE")* function - On port *($Rp == 9090)* - Add function *rtp_proxy_setup("t114t18")* above *'force_rtp_proxy()'* - On *'onreply_route[1]'* add function *rtp_proxy_setup("t18t114")*, just above *'force_rtp_proxy()'*	- "t114t18" represents transcoding parameters from AMR-NB (114) to G729 (18) - Use port 9090 to make transcoding calls - Comment out any other *rtp_proxy_setup()* function calls.
Encryption + Transcoding	- On *route[5]* - In *is_method("INVITE")* function - On port *($Rp == 9090)* - Add *rtp_proxy_setup("_enckeyt114t18")* just above *'force_rtp_proxy()'* - On *'onreply_route[1]'* add function *rtp_proxy_setup("t18t114")*, just above *'force_rtp_proxy()'*	- Both encryption and transcoding are enabled - Used port 9090 to make transcoding + encrypted calls. - Comment out any other *rtp_proxy_setup()* function calls.
All cases	Insert function '*rtp_proxy_cleanup()*' in all the above cases at - On *route[5]*, in *is_method("BYE")* function Just before *'unforce_rtp_proxy'* - In the beginning of *is_method("BYE")* function	

Notes:

1. Please restart Kamailio after any changes to the kamailio config file to factor in the changes.
2. To start Kamailio please use '*kamailio*'.
3. To check the configuration file parsing, use *'kamailio –c'* .
4. To change the logging level of kamailio edit line '*debug = n*'
   N = 1 to 7, with 1 being the lowest.

3 Signalogic Software

3.1 Compiling and Running the Encryption Engine

This section provides a brief description of the software developed by Signalogic. It also provides the list of files either modified or developed by Signalogic.

List of files developed/modified

a.) Files developed/modified in Kamailio
- i.) nathelper.c
- ii) kamailio.cfg

b.) Files developed/modified in rtpproxy
- i.) main.c
- ii.) rtpp_session.h
- iii.) rtpp_command.c
- iv.) rtplib_xced.c
- v.) rtplib_xced.h
- vi.) reconstruct_header.h
- vii.) reconstruct_header.c
- viii.) rtplib_util.c
- ix.) rtplib_util.h
- x.) rtplib_encoder.c
- xi.) rtplib_decoder.c
- xii.) Makefile.in

nathelper.c

This file is used to communicate to rtpproxy through Unix socket communication. It has been modified to send specific commands to rttproxy.

kamalio.cfg

This is the configuration script read by Kamailio on startup. This script is responsible for loading various modules and initializes various parameters. This script is modified to allow Kamailio to listen on specific ports, including a special port for encrypted data. It is also modified to call a special function in case of encryption.

main.c

This is the main file of rtpproxy. It is modified to encrypt/decrypt data, compress/decompress headers and support transcoding.

rtpp_session.h

This file defines data structures useful to maintain different rtpp sessions. It is modified to contain some additional data members.

rtpp_command.c
This source is responsible for implementing functions to parse and process various RTP commands. It has been modified to add certain new commands.

rtplib_xced.c and **rtplib_xced.h**

These file contains the support for encryption and decryption algorithms.
They also contain support for transcoding. Currently transcoding from AMR-NB to G729 and vice versa

reconstruct_header.h and **reconstruct_header.c**

These files contain function to support RTP header compression and decompression.

rtplib_encoder.c and **rtplib_decoder.c**

These files support AMR-NB and G729 encoder and decoder functions. Theses files provide the base for transcoding functionality.

Makefile.in

This file is used to add Signalogic specific libraries, header files and source files in the Makefile used to build rtpproxy.

4 Test Call Setup

4.1 Scenario 1 (X-lite to X-lite)

In order to verify Registrar functionality of Kamailio, two instances of X-lite running on separate Windows machines may be used to

(a) Register with Kamailio
(b) Make a test call

The screen captures below show example account settings for both softphones.

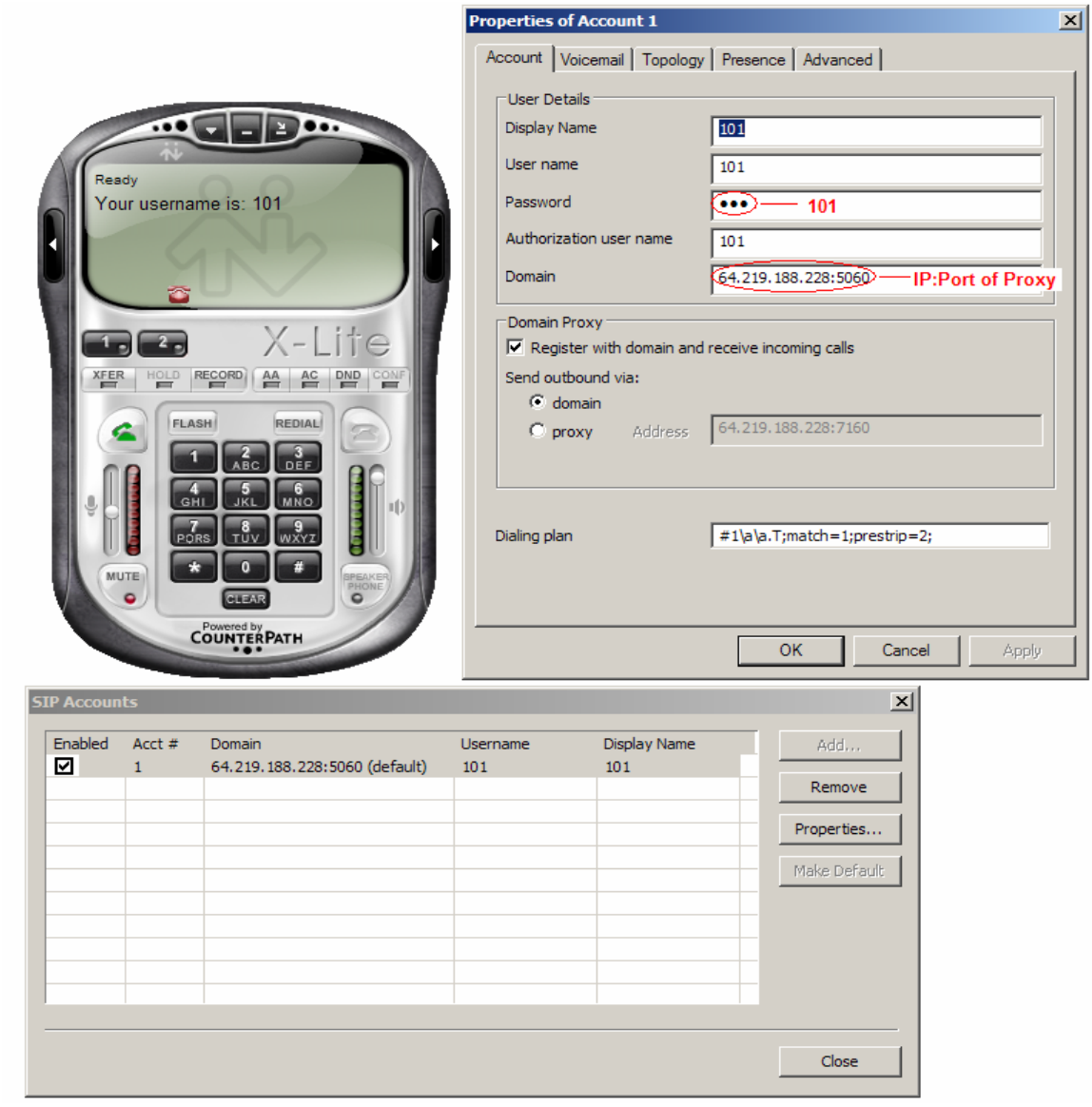

Figure 8 : Softphone 1 Account Settings

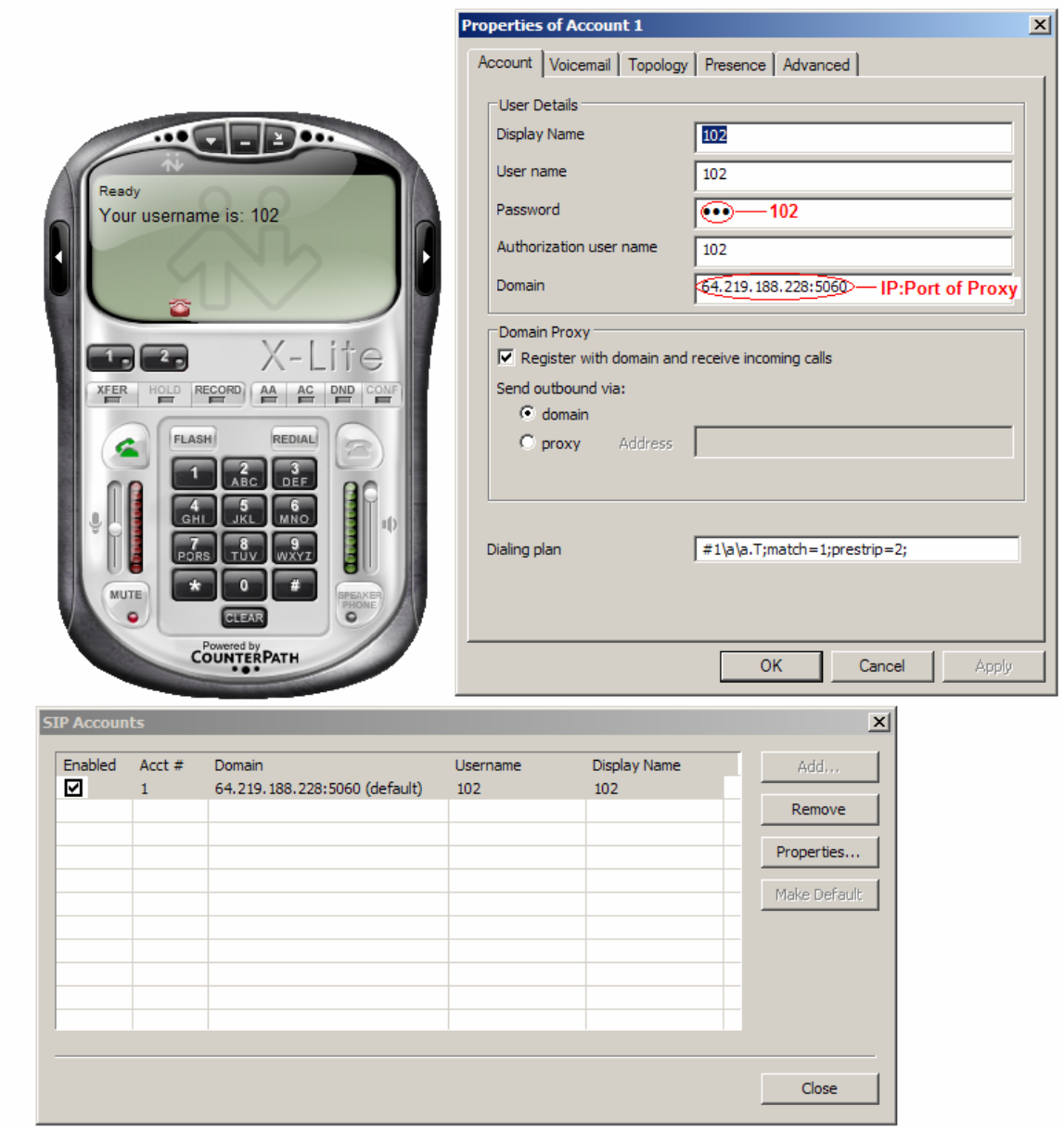

Figure 9 : Softphone 2 Account Settings

Notes

(a) X-lite softphone can be downloaded from www.counterpath.com Install both softphones on windows machines that are in the same subnet to avoid NAT issues

(b) Kamailio should be running on the proxy server, before Registration/Call tests can be carried out

(c) Once both softphones are registered, a test call can be placed

4.2 Scenario 2 (X-lite to Mobile Dialer)

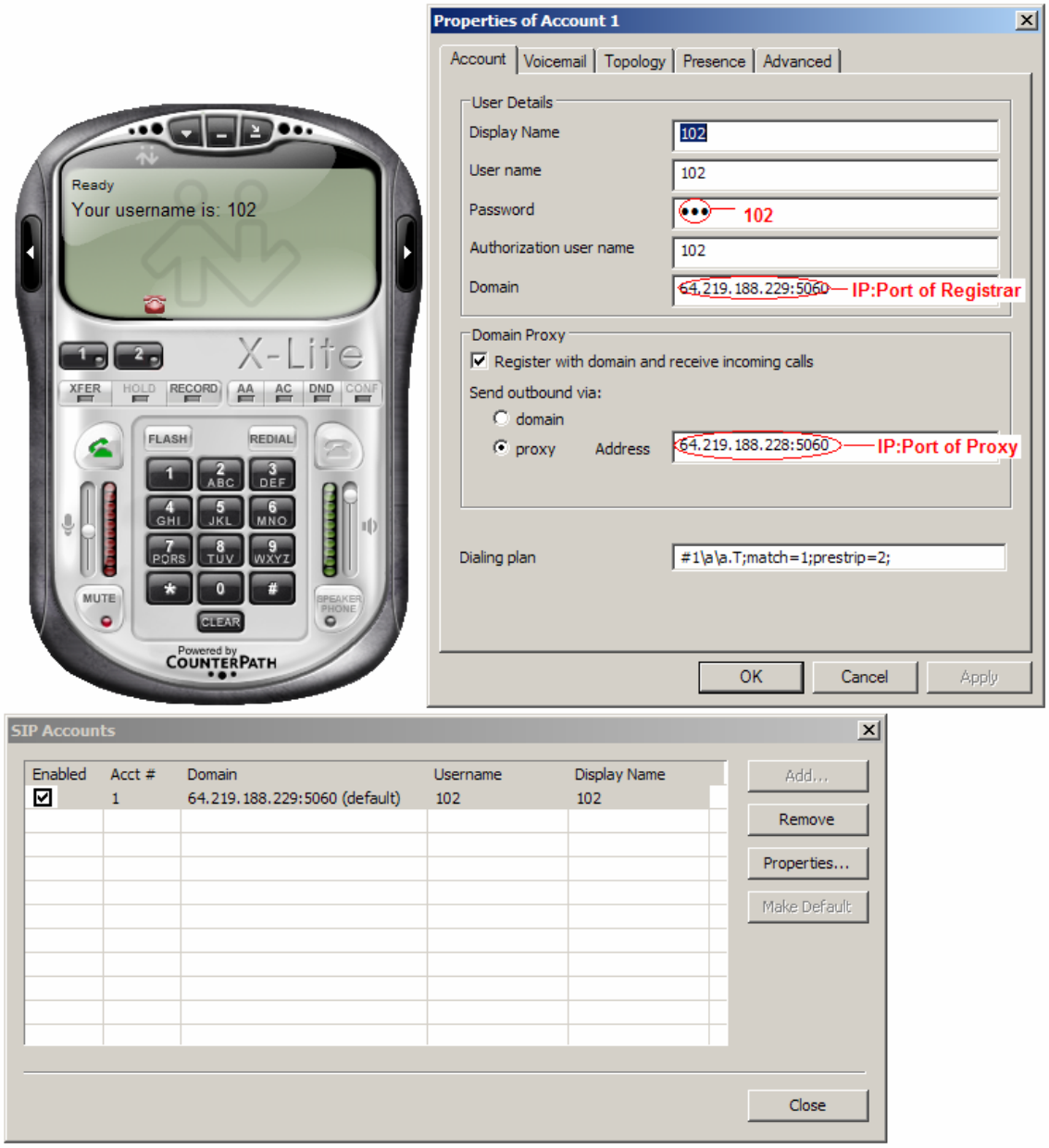

Figure 10 : Softphone Account Settings

Mobile Dialer Account Settings

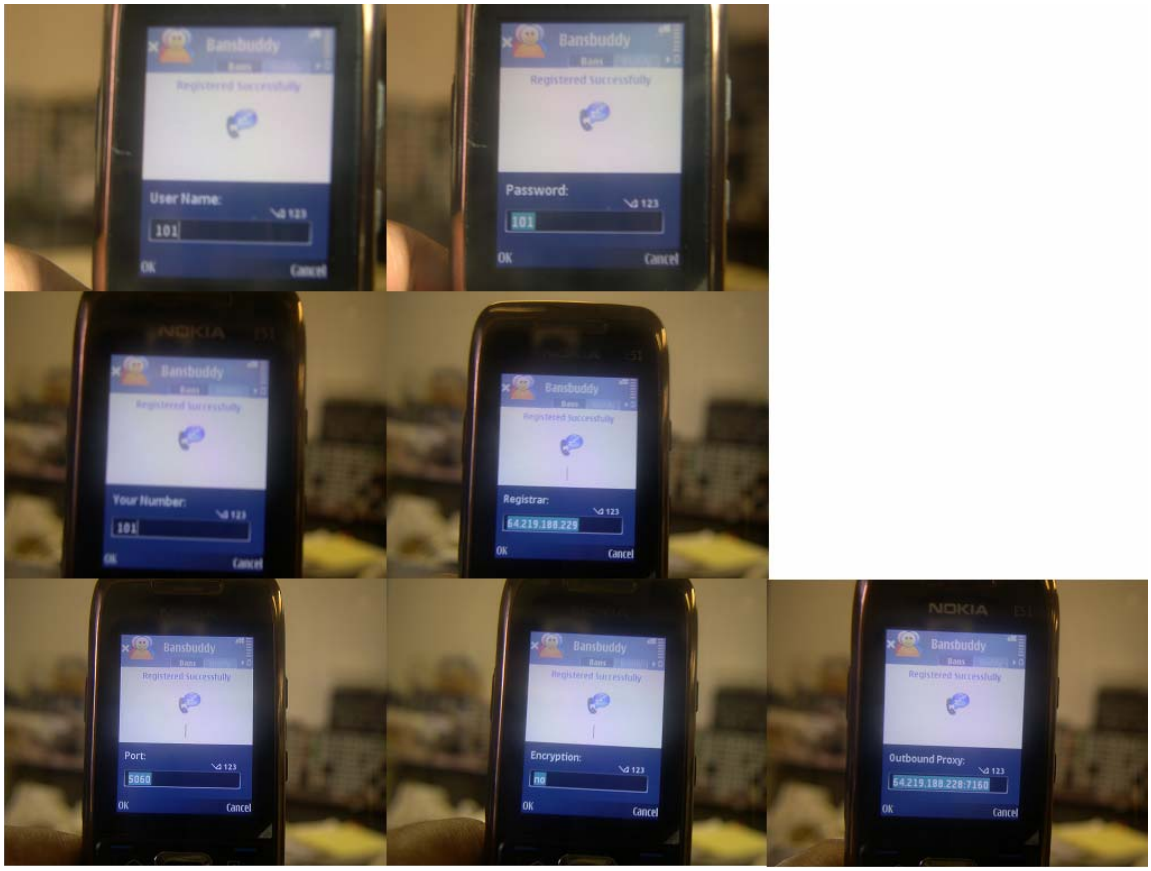

Figure 11 : Mobile dialer account configuration screens

(a) User Name : 101

(b) Password : 101

(c) Your Number : 101

(d) Registrar : 64.219.188.229

(e) Port : 5060

(f) Encryption : no (use “yes” to turn on encryption)

(g) Outbound Proxy : 64.219.188.228:7160

Notes

(a) Ensure that asterisk is running and /etc/asterisk/sip.conf contains user accounts referred to above. Example sip.conf user account section shown below

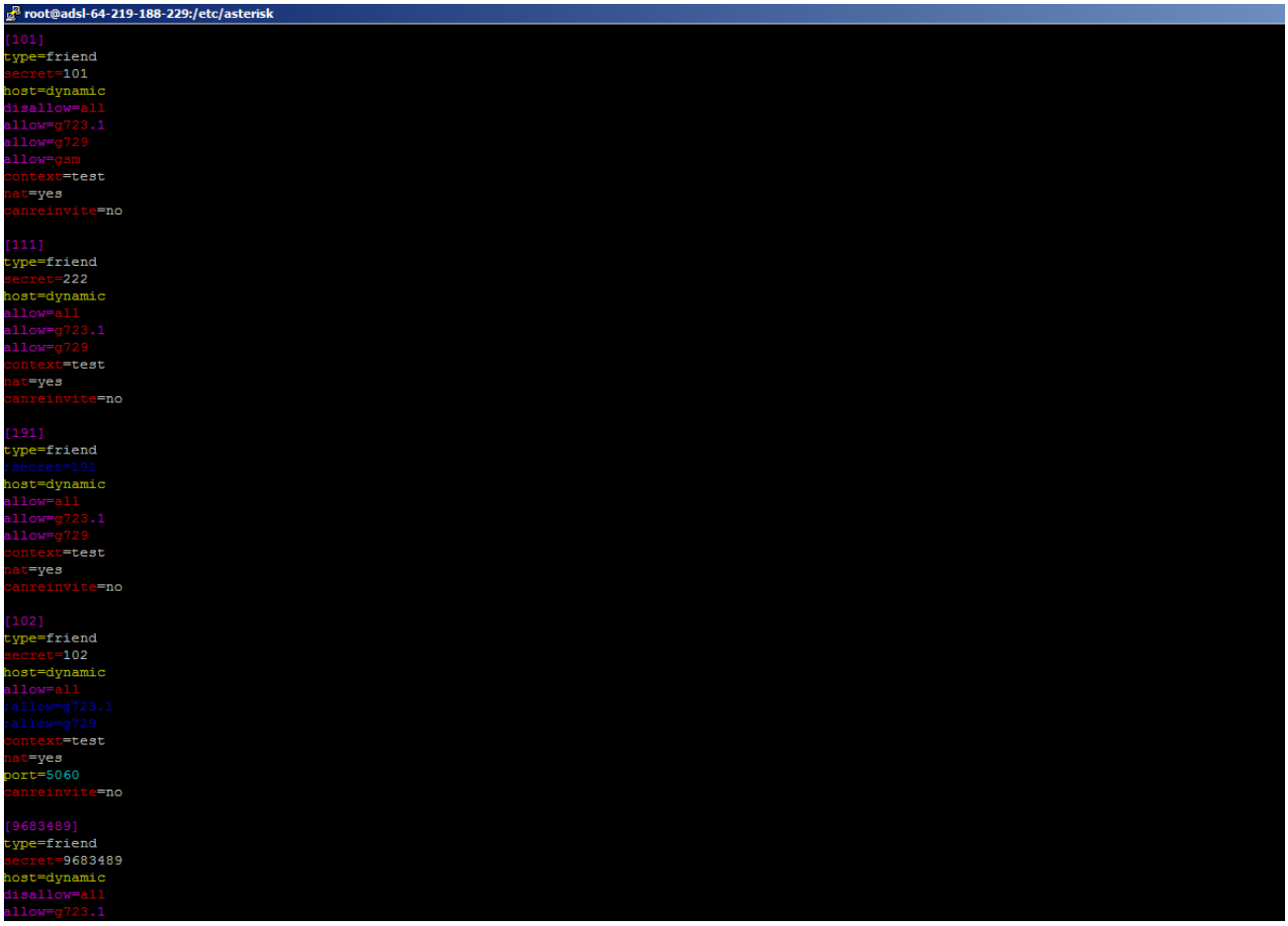

Figure 12 : Screenshot of Asterisk's sip.conf

(b) Ensure Kamailio, rtpproxy are running on the proxy server

(c) Once both devices are registered, a test call may be placed

APPENDIX A

Example Kamailio Configuration File

[Signalogic additions to the Kamailio config file are highlighted in ***bold and italic***]

```
#
# $Id: kamailio.cfg 5926 2009-09-24 18:37:43Z miconda $
#
# Kamailio (OpenSER) SIP Server - basic configuration script
#     - web: http://www.kamailio.org
#     - svn: http://openser.svn.sourceforge.net/viewvc/openser/
#
# Direct your questions about this file to: <users@lists.kamailio.org>
#
# Refer to the Core CookBook at http://www.kamailio.org/dokuwiki/doku.php
# for an explanation of possible statements, functions and parameters.
#
# There are comments showing how to enable different features in th econfig
# file. Such commented code starts with #X# where X is a letter to identify
# a feature. Delete entire #X# if you want to enable that feature. Next are
# sed commands that help you enable such features.
#
# *** To enamble mysql execute:
#     sed -i 's/#m#//g' kamailio.cfg
#
# *** To enamble authentication execute:
#     - enable mysql
#     sed -i 's/#a#//g' kamailio.cfg
#     - add users using 'kamctl'
#
# *** To enamble persistent user location execute:
#     - enable mysql
#     sed -i 's/#u#//g' kamailio.cfg
#
# *** To enamble presence server execute:
#     - enable mysql
#     sed -i 's/#p#//g' kamailio.cfg
#
# *** To enamble nat traversal execute:
#     sed -i 's/#n#//g' kamailio.cfg
#     - install rtpproxy: http://www.rtpproxy.org
#     - start rttproxy:
#        rtpproxy -l _your_public_ip_ -s udp:localhost:7722
#
# *** To enhance accounting execute:
#     - enable mysql
#     sed -i 's/#c#//g' kamailio.cfg
#     - add following columns to database
# ALTER TABLE acc ADD COLUMN src_user VARCHAR(64) NOT NULL DEFAULT '';
# ALTER TABLE acc ADD COLUMN src_domain VARCHAR(128) NOT NULL DEFAULT '';
# ALTER TABLE a 545, column cc ADD COLUMN dst_ouser VARCHAR(64) NOT NULL
DEFAULT '';
# ALTER TABLE acc ADD COLUMN dst_user VARCHAR(64) NOT NULL DEFAULT '';
# ALTER TABLE acc ADD COLUMN dst_domain VARCHAR(128) NOT NULL DEFAULT '';
```

```
# ALTER TABLE missed_calls ADD COLUMN src_user VARCHAR(64) NOT NULL DEFAULT
'';
# ALTER TABLE missed_calls ADD COLUMN src_domain VARCHAR(128) NOT NULL
DEFAULT '';
# ALTER TABLE missed_calls ADD COLUMN dst_ouser VARCHAR(64) NOT NULL DEFAULT
'';
# ALTER TABLE missed_calls ADD COLUMN dst_user VARCHAR(64) NOT NULL DEFAULT
'';
# ALTER TABLE missed_calls ADD COLUMN dst_domain VARCHAR(128) NOT NULL
DEFAULT '';
#


####### Global Parameters #########

debug=6
log_stderror=no
log_facility=LOG_LOCAL0

fork=yes
children=4

/* uncomment the following lines to enable debugging */
#debug=6
#fork=no
#log_stderror=yes
/* uncomment the next line to disable TCP (default on) */
#disable_tcp=yes

/* uncomment the next line to enable the auto temporary blacklisting of
   not available destinations (default disabled) */
#disable_dns_blacklist=no

/* uncomment the next line to enable IPv6 lookup after IPv4 dns
   lookup failures (default disabled) */
#dns_try_ipv6=yes

/* uncomment the next line to disable the auto discovery of local aliases
   based on revers DNS on IPs (default on) */
#auto_aliases=no

/* uncomment the following lines to enable TLS support  (default off) */
#disable_tls = no
#tls_verify_server = 1
#tls_verify_client = 0
#tls_require_client_certificate = 0
#tls_method = TLSv1
#tls_certificate = "/usr/local/etc/kamailio/tls/user/user-cert.pem"
#tls_private_key = "/usr/local/etc/kamailio/tls/user/user-privkey.pem"
#tls_ca_list     = "/usr/local/etc/kamailio/tls/user/user-calist.pem"


#port=7160

/* uncomment and configure the following line if you want Kamailio to
   bind on a specific interface/port/proto (default bind on all available) */
```

```
#Following 2 listening IP:port's are used for unencrypted calls only
listen=udp:64.219.188.228:5060
listen=udp:64.219.188.228:7160

# Following listening IP:port is used for encrypted calls only
listen=udp:64.219.188.228:9090

####### Modules Section ########

#set module path
mpath="/usr/local/lib/kamailio/modules_k/:/usr/local/lib/kamailio/modules/"

/* uncomment next line for MySQL DB support */
#loadmodule "db_mysql.so"
loadmodule "mi_fifo.so"
loadmodule "kex.so"
loadmodule "tm.so"
loadmodule "tmx.so"
loadmodule "sl.so"
loadmodule "rr.so"
loadmodule "pv.so"
loadmodule "maxfwd.so"
loadmodule "usrloc.so"
loadmodule "registrar.so"
loadmodule "textops.so"
#loadmodule "uri_db.so"
loadmodule "siputils.so"
loadmodule "xlog.so"
loadmodule "sanity.so"
loadmodule "ctl.so"
loadmodule "mi_rpc.so"
loadmodule "acc.so"
#loadmodule "sig.so"           /*  Added by Pranav */
/* uncomment next lines for MySQL based authentication support
   NOTE: a DB (like db_mysql) module must be also loaded */
#loadmodule "auth.so"

#loadmodule "auth_db.so"
/* uncomment next line for aliases support
   NOTE: a DB (like db_mysql) module must be also loaded */
#loadmodule "alias_db.so"
/* uncomment next line for multi-domain support
   NOTE: a DB (like db_mysql) module must be also loaded
   NOTE: be sure and enable multi-domain support in all used modules
         (see "multi-module params" section ) */
#loadmodule "domain.so"
/* uncomment the next two lines for presence server support
   NOTE: a DB (like db_mysql) module must be also loaded */
#p#loadmodule "presence.so"
#p#loadmodule "presence_xml.so"

loadmodule "nathelper.so"

# ----------------- setting module-specific parameters ---------------

# ----- mi_fifo params -----
```

```
modparam("mi_fifo", "fifo_name", "/tmp/kamailio_fifo")


# ----- rr params -----

# add value to ;lr param to cope with most of the UAs
modparam("rr", "enable_full_lr", 1)
# do not append from tag to the RR (no need for this script)
modparam("rr", "append_fromtag", 0)


# ----- rr params -----
modparam("registrar", "method_filtering", 1)
/* uncomment the next line to disable parallel forking via location */
# modparam("registrar", "append_branches", 1)
/* uncomment the next line not to allow more than 10 contacts per AOR */
#modparam("registrar", "max_contacts", 10)


# ----- uri_db params -----
/* by default we disable the DB support in the module as we do not need it
   in this configuration */
#modparam("uri_db", "use_uri_table", 0)
#modparam("uri_db", "db_url", "")


# ----- acc params -----
/* what sepcial events should be accounte
d ? */
modparam("acc", "early_media", 1)
modparam("acc", "report_ack", 1)
modparam("acc", "report_cancels", 1)
/* by default ww do not adjust the direct of the sequential requests.
   if you enable this parameter, be sure the enable "append_fromtag"
   in "rr" module */
modparam("acc", "detect_direction", 0)
/* account triggers (flags) */
modparam("acc", "failed_transaction_flag", 3)
modparam("acc", "log_flag", 1)
modparam("acc", "log_missed_flag", 2)
modparam("acc", "log_extra",

"src_user=$fU;src_domain=$fd;dst_ouser=$tU;dst_user=$rU;dst_domain=$rd")
/* uncomment the following lines to enable DB accounting also */
#modparam("acc", "db_flag", 1)
#modparam("acc", "db_missed_flag", 2)
#modparam("acc", "db_url",
#        "mysql://openser:openserrw@localhost/openser")
#modparam("acc", "db_extra",
#
"src_user=$fU;src_domain=$fd;dst_ouser=$tU;dst_user=$rU;dst_domain=$rd")


# ----- usrloc params -----

/* uncomment the following lines if you want to enable DB persistency
   for location entries */
```

```
#modparam("usrloc", "db_mode",   2)
#modparam("usrloc", "db_url",
#        "mysql://openser:openserrw@localhost/openser")

# ----- auth_db params -----
/* uncomment the following lines if you want to enable the DB based
   authentication */
#modparam("auth_db", "calculate_ha1", yes)
#modparam("auth_db", "password_column", "password")
#modparam("auth_db", "db_url",
#        "mysql://openser:openserrw@localhost/openser")
#modparam("auth_db", "load_credentials", "")


# ----- alias_db params -----
/* uncomment the following lines if you want to enable the DB based
   aliases */
#modparam("alias_db", "db_url",
#        "mysql://openser:openserrw@localhost/openser")


# ----- domain params -----
/* uncomment the following lines to enable multi-domain detection
   support */
#modparam("domain", "db_url",
#       "mysql://openser:openserrw@localhost/openser")
#modparam("domain", "db_mode", 1)   # Use caching


# ----- multi-module params -----
/* uncomment the following line if you want to enable multi-domain support
   in the modules (dafault off) */
#modparam("alias_db|auth_db|usrloc|uri_db", "use_domain", 1)


# ----- presence params -----
/* uncomment the following lines if you want to enable presence */
#p#modparam("presence|presence_xml", "db_url",
#p#     "mysql://openser:openserrw@localhost/openser")
#p#modparam("presence_xml", "force_active", 1)
#p#modparam("presence", "server_address", "sip:192.168.1.2:5060")

# -- nathelper
modparam("nathelper", "rtpproxy_sock", "udp:127.0.0.1:7722")
modparam("nathelper", "natping_interval", 30)
modparam("nathelper", "ping_nated_only", 1)
modparam("nathelper", "sipping_bflag", 7)
modparam("nathelper", "sipping_from", "sip:pinger@kamailio.org")
#  -- nathelper signalogic parameter
modparam("nathelper", "signalogic_extensions_enable", 1)
modparam("registrar|nathelper", "received_avp", "$avp(i:80)")
modparam("usrloc", "nat_bflag", 6)

####### Routing Logic ########


# main request routing logic
```

```
route{

      xlog("L_DBG","SIP message was received on port [$Rp]\n");

        if (!mf_process_maxfwd_header("10")) {
                sl_send_reply("483","Too Many Hops");
                exit;
        }

        # NAT detection
        route(4);

        if (has_totag()) {
                # sequential request withing a dialog should
                # take the path determined by record-routing
                if (loose_route()) {
                        if (is_method("BYE")) {
                                setflag(1); # do accounting ...
                                setflag(3); # ... even if the transaction
fails
                        }
                        route(1);
                } else {
                        if (is_method("SUBSCRIBE") && uri == myself) {
                                # in-dialog subscribe requests
                                route(2);
                                exit;
                        }
                        if ( is_method("ACK") ) {
                                if ( t_check_trans() ) {
                                        # non loose-route, but stateful ACK;
must be an ACK after a 487 or e.g. 404 from upstream server
                                        t_relay();
                                        start_recording();
                                        exit;
                                } else {
                                        # ACK without matching transaction ...
ignore and discard.\n");
                                        exit;
                                }
                        }
                        sl_send_reply("404","Not here");
                }
                exit;
        }

        #initial requests

        # CANCEL processing
        if (is_method("CANCEL"))
        {

            rtp_proxy_cleanup();
                if (t_check_trans())
                        t_relay();
                exit;
```

```
        }


        t_check_trans();

        # authentication
        route(3);

        # record routing
        if (!is_method("REGISTER|MESSAGE"))
                record_route();
        # account only INVITEs
        if ($fU==":"&&is_method("INVITE")) {
#       if(is_method("INVITE")) {
                setflag(1); # do accounting
        record_route();
#        rewritehost("udp:64.219.188.229:7160");
        t_relay();

        }
        if (!uri==myself)

        /* replace with following line if multi-domain support is used */
        ##if (!is_uri_host_local())
        {
                append_hf("P-hint: outbound\r\n");
                # if you have some interdomain connections via TLS
                ##if($rd=="tls_domain1.net") {
                ##      t_relay("tls:domain1.net");
                ##      exit;
                ##} else if($rd=="tls_domain2.net") {
                ##      t_relay("tls:domain2.net");
                ##      exit;
                ##}
                route(1);
        }

        # requests for my domain

        if( is_method("PUBLISH|SUBSCRIBE"))
                route(2);

        if (is_method("REGISTER"))
        {
#                if (isflagset(5)) {
#n#                      setbflag(6);
#n#                      setbflag(7);
#                }
                if (!save("location"))
                        sl_reply_error();


                exit;
        }


        if ($rU==$null) {
```

```
                # request with no Username in RURI
                sl_send_reply("484","Address Incomplete");
                exit;
        }

        # apply DB based aliases (uncomment to enable)
#        alias_db_lookup("dbaliases");

        if (!lookup("location")) {
                switch ($retcode) {
                        case -1:
                        case -3:
                                t_newtran();
                                t_reply("404", "Not Found");
                                exit;
                        case -2:
                                sl_send_reply("405", "Method Not Allowed");
                                exit;

                }
        }

        # when routing via usrloc, log the missed calls also
        setflag(2);
        route(1);
}


route[1] {
        if (check_route_param("nat=yes")) {
                setbflag(6);
        }
if (isflagset(5) || isbflagset(6)) {
                route(5);
        }

        /* example how to enable some additional event routes */
        if (is_method("INVITE")) {
#Attempting textops example
# replace(" 114 ","");
#End attempt
#               t_on_branch("1");

                t_on_reply("1");
                t_on_failure("1");
        }

        if (!t_relay()) {
                sl_reply_error();
        }
        exit;
}


# Presence route
/* uncomment the whole following route for enabling presence server */
route[2]
```

```
{
#p#     if (!t_newtran())
#p#     {
#p#             sl_reply_error();
#p#             exit;
#p#     };
#p#
#p#     if(is_method("PUBLISH"))
#p#     {REGISTER
#p#             handle_publish();
#p#             t_release();
#p#     }
#p#     else
#p#     if( is_method("SUBSCRIBE"))
#p#     {
#p#             handle_subscribe();
#p#             t_release();
#p#     }
#p#     exit;

        # if presence enabled, this part will not be executed
        if (is_method("PUBLISH") || $rU==$null)
        {
                sl_send_reply("404", "Not here");
                exit;
        }
        return;
}

# Authentication route
/* uncomment the whole following route for enabling authentication */
route[3] {
#       if (is_method("REGISTER"))
#       {
                # authenticate the REGISTER requests (uncomment to enable
auth)
#               if (!www_authorize("", "subscriber"))
#               {
#                       www_challenge("", "0");
#                       exit;
#               }
#a#
#               if ($au!=$tU)
#               {
#                       sl_send_reply("403","Forbidden auth ID");
#                       exit;
#               }
#       } else {
#a#             # authenticate if from local subscriber (uncomment to enable
auth)
#               if (from_uri==myself)

#               {
#                       if (!proxy_authorize("", "subscriber")) {
#                               proxy_challenge("", "0");
#                               exit;ne 53
#               }
```

```
#                       if (is_method("PUBLISH"))
#                       {
#                               if ($au!=$tU) {
#                                       sl_send_reply("403","Forbidden auth
ID");
#                                       exit;
#                               }
#                       } else {
#                               if ($au!=$fU) {
#                               sl_send_reply("403","Forbidden auth ID");
#                                       exit;
#                               }
#                       }
#a#
#                       consume_credentials();
#a#                     # caller authenticated
#               }
#       }

        return;
}

# Caller NAT detection route
/* uncomment the whole following route for enabling Caller NAT Detection */
route[4]{
        force_rport();
#        if (method=="REGISTER")&&($fU==":.") {
              if (method=="REGISTER") {
           xlog("L_DBG","Calling fix_nated_register");
               fix_nated_register();
        } else {
           xlog("L_DBG","Calling fix_nated_contact");
               fix_nated_contact();
        }
        setflag(5);
        return;
}

# rtpproxy control
/* uncomment the whole following route for enabling rtpproxy Control */
route[5] {
#       if (is_method("BYE")&&$fU==":.") {
        if (is_method("BYE")) {
           rtp_proxy_cleanup();
           unforce_rtp_proxy();
        } else if (is_method("INVITE")){
           if ($Rp == 9090){
              if($si =~ “(SipServerIP1|SipServerIP2|...)”) {
                 rtp_proxy_setup(“_”);
              }else {
                rtp_proxy_setup(“_encryptionkey”);
            }
           }else{
                    xlog("L_DBG","Calling force_rtp_proxy with Only
transcoding mode \n");
#                       if (!rtp_proxy_setup("t114t18")){
#                       sl_send_reply("480","Temporarily Unavailable");
```

```
#                       exit;
#                   }
            }
                  force_rtp_proxy();

            }
        if (!has_totag()) add_rr_param(";nat=yes");
        return;
}

branch_route[1] {
        xdbg("new branch at $ru\n");
}
onreply_route[1] {
        xdbg("incoming reply\n");
        if ((isflagset(5) || isbflagset(6)) && status=~"(183)|180|(2[0-9][0-
9])") {
             if ($Rp == 9090){
              if($si =~ "(SipServerIP1|SipServerIP2|...)") {
                 rtp_proxy_setup("_");
              }else {
                 rtp_proxy_setup("_encryptionkey");
              }
             }
      }
           force_rtp_proxy("z120");

        if (isbflagset(6)) {
              fix_nated_contact();
        }
}

failure_route[1] {
       if(status =~ "400|404|408|486|302|407"){
          rtp_proxy_cleanup();
       }
       if (is_method("INVITE") && (isbflagset(6) || isflagset(5))) {
             unforce_rtp_proxy();
       }

#        if (t_is_cancelled()) {
#               exit;
#        }

        # uncomment the following lines if you want to block client
        # redirect based on 3xx replies.
        ##if (t_check_status("3[0-9][0-9]")) {
        ##t_reply("404force_rtp_proxy","Not found");
        ##      exit;
        ##}

        # uncomment the following lines if you want to redirect the failed
        # calls to a different new destination
        ##if (t_check_status("486|408")) {
        ##      sethostport("192.168.2.100:5060");
        ##      append_branch();
        ##      # do not set the missed call flag again
```

```
        ##      t_relay();
        ##}
}
```

APPENDIX B

Independent control over encryption of SIP, Media

The default configuration of Kamailio and rtpproxy as mentioned in Section 2 of this document results in a system where both SIP and Media are encrypted.

Independent control may be exercised over encryption of SIP, Media such that SIP-only encrypted or Media-only encrypted calls may be made from the dialer. Such control may be enforced through modifications to two system components, namely

(a) Iptables rules
(b) Kamailio configuration file

The following two sections explain how these components may be modified depending on whether SIP-only encryption or Media-only encryption is desired.

1) SIP-only Encryption

To add capabilities for SIP-only encryption to a proxy that has a default configuration of Kamailio and rtpproxy as described in Section 2 of this document, the following changes need to be made

(a) Add the following Iptables rules

```
/usr/sbin/iptables –t mangle –A PREROUTING –p udp --dport 8000 –j NFQUEUE
/usr/sbin/iptables –t magle –A OUTPUT –p udp --sport 8000 –j NFQUEUE
```

(b) Add to kamailio.cfg, listening IP address corresponding to calls with SIP-only encryption

```
listen=udp:proxy_ip_addr:8000
```

Notes:

- Port number 8000 is chosen for illustration purposes only

- In the example, an additional UDP port 8000 is used, however there is no restriction on the transport protocol, i.e. both UDP and TCP may be used

- The underlying assumption is that all SIP-only encrypted calls will be routed through port 8000 (illustration purposes only) of the proxy

2) Media-only Encryption

To add capabilities for Media-only encryption to a proxy that has a default configuration of Kamailio and rtpproxy as described in Section 2 of this document, the following changes need to be made

Modify kamailio.cfg as shown below

- Add a listening IP address corresponding to calls with Media-only encryption

  listen= udp:proxy_ip_addr:8000

- Modify the ***if*** statements in **route[5]** and **onreply_route[1]** as shown below

```
route[5] {
if (is_method("BYE")) {
      rtp_proxy_cleanup();
      unforce_rtp_proxy();
  } else if (is_method("INVITE")){
if ($Rp == 9090 || $Rp == 8000){    ### New port added here ###
         if($si =~ "(SipServerIP1|SipServerIP2|...)") {
            rtp_proxy_setup("_");
         }else {
           rtp_proxy_setup("_encryptionkey");
       }
 }
force_rtp_proxy();

onreply_route[1] {
xdbg("incoming reply\n");
if ((isflagset(5)||isbflagset(6))&&status=~"(183)|180|(2[0-9][0-9])"){
    if ($Rp == 9090 || $Rp == 8000){ ### New port added here ###
       if($si =~ "(SipServerIP1|SipServerIP2|...)") {
          rtp_proxy_setup("_");
       }else {
          rtp_proxy_setup("_encryptionkey");
       }
      }
}
force_rtp_proxy("z120");
```

Notes:

- Port number 8000 is chosen for illustration purposes only

- In the example, an additional UDP port 8000 is used, however there is no restriction on the transport protocol, i.e. both UDP and TCP may be used

- The underlying assumption is that all Media-only encrypted calls will be routed through port 8000 (illustration purposes only) of the proxy